@file:OptIn(ExperimentalMaterial3Api::class)  
  
package com.example.myapplication  
  
import android.content.Context  
import android.os.Bundle  
import android.util.Log  
import android.widget.Toast  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.PaddingValues  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.Spacer  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.fillMaxWidth  
import androidx.compose.foundation.layout.height  
import androidx.compose.foundation.layout.padding  
import androidx.compose.foundation.layout.size  
import androidx.compose.foundation.layout.width  
import androidx.compose.foundation.lazy.LazyColumn  
import androidx.compose.foundation.lazy.items  
import androidx.compose.foundation.shape.*CircleShape*import androidx.compose.foundation.shape.RoundedCornerShape  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.*Add*import androidx.compose.material3.Button  
import androidx.compose.material3.ButtonDefaults  
import androidx.compose.material3.ExperimentalMaterial3Api  
import androidx.compose.material3.ExtendedFloatingActionButton  
import androidx.compose.material3.FloatingActionButton  
import androidx.compose.material3.Icon  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.ModalBottomSheet  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.Surface  
import androidx.compose.material3.Text  
import androidx.compose.material3.TextField  
import androidx.compose.material3.TextFieldColors  
import androidx.compose.material3.TextFieldDefaults  
import androidx.compose.material3.TopAppBar  
import androidx.compose.material3.TopAppBarDefaults  
import androidx.compose.material3.rememberModalBottomSheetState  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.LaunchedEffect  
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.mutableStateListOf  
import androidx.compose.runtime.mutableStateOf  
import androidx.compose.runtime.remember  
import androidx.compose.runtime.rememberCoroutineScope  
import androidx.compose.runtime.setValue  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.draw.clip  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.graphics.vector.ImageVector  
import androidx.compose.ui.layout.ContentScale  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.res.painterResource  
import androidx.compose.ui.res.vectorResource  
import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.unit.sp  
import com.example.myapplication.ui.theme.MyApplicationTheme  
import com.google.gson.Gson  
import kotlinx.coroutines.launch  
import com.example.myapplication.AppointmentCard as AppointmentCard  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** MyApplicationTheme **{** // A surface container using the 'background' color from the theme  
 Surface(  
 ) **{** val name = ""  
 val position = ""  
 val photoLink = ""  
 val date = ""  
 val time = ""  
 ScaffoldExample(name, position, photoLink, date, time)  
 **}  
  
 }  
 }** }  
}  
data class AppointmentCardModel(  
 val name: String,  
 val position: String,  
 val photoLink: String,  
 val date: String,  
 val time: String  
)  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun ScaffoldExample(name: String, position: String, photoLink: String, data: String, time: String) {  
  
 val context = *LocalContext*.current  
 val sheredPreferens = remember **{** context.getSharedPreferences("main", Context.*MODE\_PRIVATE*)  
 **}** val appointmentDataList = remember **{** *mutableStateListOf*<AppointmentCardModel>() **}** LaunchedEffect(Unit) **{** val jsonData = sheredPreferens.getString("cardModels", null)?: return@LaunchedEffect  
 val gson = Gson()  
 val data = gson.fromJson(jsonData, Array<AppointmentCardModel>::class.*java*).*toList*()  
 appointmentDataList.addAll(data)  
 **}** val addAppointmentBottomSheetVisible = remember **{** *mutableStateOf*(false)  
 **}** Scaffold(  
 topBar = **{** TopAppBar(  
 title = **{** Text(  
 text = "Some text",  
 color = *Color*(0xFFFFFFFF)  
 )  
 **}**,  
 colors = TopAppBarDefaults.topAppBarColors(  
 containerColor = *Color*(0xFF4894FE)  
 )  
 )  
 **}**,  
 floatingActionButton = **{** FloatingActionButton(  
 containerColor = *Color*(0xFFECE6F0),  
 onClick = **{** addAppointmentBottomSheetVisible.value = true **}**) **{** Icon(  
 imageVector = Icons.Filled.*Add*,  
 contentDescription = null  
 )  
 **}  
 }** ) **{** paddingValues **->** LazyColumn(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(paddingValues),  
 contentPadding = *PaddingValues*(all = 30.*dp*),  
 verticalArrangement = Arrangement.spacedBy(30.*dp*)  
 ) **{** *items*(items = appointmentDataList) **{** data **->** AppointmentCard(  
 name = data.name,  
 position = data.position,  
 photoLink = data.photoLink,  
 date=data.date,  
 time = data.time  
 )  
 **}  
 }** if (addAppointmentBottomSheetVisible.value) {  
 ModalBottomSheet(  
 onDismissRequest = **{** addAppointmentBottomSheetVisible.value = false **}**,  
 containerColor = Color.White,  
 content = **{** val name = remember**{***mutableStateOf*("")**}** val position = remember**{***mutableStateOf*("")**}** val date = remember**{***mutableStateOf*("")**}** val recordingTime = remember**{***mutableStateOf*("")**}** TextField(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(horizontal = 30.*dp*),  
 value = name.value,  
 label = **{** Text(text = "Имя и фамилия") **}**,  
 onValueChange = **{**name.value = **it}**,  
 colors = TextFieldDefaults.colors(  
 focusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 unfocusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 disabledContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 errorContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 )  
 )  
 Spacer(  
 modifier = Modifier.*height*(12.*dp*)  
 )  
 TextField(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(horizontal = 30.*dp*),  
 value = position.value,  
 label = **{** Text(text = "Должность") **}**,  
 onValueChange = **{**position.value = **it}**,  
 colors = TextFieldDefaults.colors(  
 focusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 unfocusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 disabledContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 errorContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 )  
 )  
 Spacer(  
 modifier = Modifier.*height*(12.*dp*)  
 )  
 TextField(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(horizontal = 30.*dp*),  
 value = date.value,  
 label = **{** Text(text = "Дата записи") **}**,  
 onValueChange = **{**date.value = **it}**,  
 colors = TextFieldDefaults.colors(  
 focusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 unfocusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 disabledContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 errorContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 )  
 )  
 Spacer(  
 modifier = Modifier.*height*(12.*dp*)  
 )  
 TextField(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(horizontal = 30.*dp*),  
 value = recordingTime.value,  
 label = **{** Text(text = "Время записи") **}**,  
 onValueChange = **{** recordingTime.value = **it}**,  
 colors = TextFieldDefaults.colors(  
 focusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 unfocusedContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 disabledContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 errorContainerColor = *Color*(0xFF4894FE).copy(0.2f),  
 )  
 )  
 Spacer(  
 modifier = Modifier.*height*(40.*dp*)  
 )  
 Button(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(  
 start = 80.*dp*,  
 end = 80.*dp*,  
 bottom = 56.*dp* ),  
 colors = ButtonDefaults.buttonColors(  
 containerColor = *Color*(0xFF4894FE)  
 ),  
 onClick = **{** val gson = Gson()  
 val cardModel = AppointmentCardModel(  
 name = name.value,  
 position =position.value ,  
 photoLink = "",  
 date = date.value,  
 time = recordingTime.value  
 )  
 val currentValueJson = sheredPreferens.getString("cardModels",null)  
 val listToWrite = if (currentValueJson == null) {  
 *arrayOf*(cardModel)  
 } else {  
 val mutableList = gson.fromJson(currentValueJson, Array<AppointmentCardModel>::class.*java*).*toMutableList*()  
 mutableList.add(cardModel)  
 mutableList.*toTypedArray*()  
 }  
 val jsonOutput = gson.toJson(listToWrite)  
 sheredPreferens.edit().putString("cardModels", jsonOutput).apply()  
  
 appointmentDataList.add(cardModel)  
 addAppointmentBottomSheetVisible.value = false  
 name.value = ""  
 position.value = ""  
 date.value = ""  
 recordingTime.value = ""  
 **}** ) **{** Text(text = "Добавить")  
 **}  
 }** )  
 }  
 **}**}  
  
@Composable  
fun AppointmentCard(  
 name: String,  
 position: String,  
 photoLink: String,  
 date: String,  
 time: String  
) {  
 Column(  
 modifier = Modifier  
 .*background*(  
 color = *Color*(0xFF4894FE),  
 shape = *RoundedCornerShape*(12.*dp*)  
 )  
 .*padding*(all = 20.*dp*)  
 ) **{** Row(  
 modifier = Modifier.*fillMaxWidth*(),  
 verticalAlignment = Alignment.CenterVertically,  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** Row(  
 verticalAlignment = Alignment.CenterVertically  
 ) **{** Image(  
 painter = painterResource(id = R.drawable.*image*),  
 contentDescription = null,  
 contentScale = ContentScale.Crop,  
 modifier = Modifier  
 .*size*(48.*dp*)  
 .*clip*(*CircleShape*)  
 )  
 Spacer(  
 modifier = Modifier.*width*(12.*dp*)  
 )  
 Column **{** Text(  
 text = name,  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Bold,  
 color = Color.White  
 )  
 Spacer(  
 modifier = Modifier.*height*(8.*dp*)  
 )  
 Text(  
 text = position,  
 fontSize = 14.*sp*,  
 fontWeight = FontWeight.Normal,  
 color = *Color*(0xFFCBE1FF)  
 )  
 **}  
 }** Icon(  
 imageVector = ImageVector.vectorResource(id = R.drawable.*ic\_arrow\_right*),  
 tint = Color.White,  
 contentDescription = null,  
 )  
 **}** Spacer(  
 modifier = Modifier.*height*(16.*dp*)  
 )  
 Box(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*height*(1.*dp*)  
 .*background*(Color.White.copy(alpha = 0.15f))  
 )  
 Spacer(  
 modifier = Modifier.*height*(16.*dp*)  
 )  
 Row **{** Row **{** Icon(  
 imageVector = ImageVector.vectorResource(id = R.drawable.*calendar*),  
 tint = Color.White,  
 contentDescription = null,  
 modifier = Modifier.*size*(16.*dp*)  
 )  
 Spacer(  
 modifier = Modifier.*width*(8.*dp*)  
 )  
 Text(  
 text = date,  
 fontSize = 12.*sp*,  
 color = Color.White,  
 fontWeight = FontWeight.Normal  
 )  
 **}** Spacer(  
 modifier = Modifier.*width*(34.*dp*)  
 )  
 Row **{** Icon(  
 imageVector = ImageVector.vectorResource(id = R.drawable.*clock*),  
 tint = Color.White,  
 contentDescription = null,  
 modifier = Modifier.*size*(16.*dp*)  
 )  
 Spacer(  
 modifier = Modifier.*width*(8.*dp*)  
 )  
 Text(  
 text = time,  
 fontSize = 12.*sp*,  
 color = Color.White,  
 fontWeight = FontWeight.Normal  
 )  
 **}  
 }  
 }**}  
  
@Preview  
@Composable  
fun ScaffoldExamplePreview() {  
 AppointmentCard(  
 name= "0",  
 position= "0",  
 photoLink= "0",  
 date= "0",  
 time= "0"  
 )  
}